View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by CERN Document Server

TUTORIAL ON QUASI-SPARSE EIGENVECTOR DIAGONALIZATION

DEAN LEE
University of Massachusetts, Amherst, MA 01003 USA
dlee@physics.umass.edu

We review several topics related to the diagonalization of quantum field Hamiltonians using the quasi-sparse eigenvector (QSE) method.

1 Introduction

Quasi-sparse eigenvector (QSE) diagonalization is a new computational method which finds the low-lying eigenvalues and eigenvectors for a general quantum field Hamiltonian [1]. It is able to handle the exponential increase in the size of Fock space for large systems by exploiting the sparsity of the Hamiltonian. QSE diagonalization can even be applied directly to infinite-dimensional systems. The method is most effective when the splitting between low-lying eigenvalues is not too small compared to the size of the off-diagonal Hamiltonian matrix entries. In such cases the low-lying eigenvectors are quasi-sparse, meaning that the vector is dominated by a small fraction of its largest components. The QSE algorithm can then be applied to the Hamiltonian H using the following steps:

1. Select a subset of orthonormal basis vectors $\{e_{i_1}, \cdots, e_{i_n}\}$ and call the corresponding subspace S.
2. Diagonalize H restricted to S and find one eigenvector v.
3. Sort the basis components $\langle e_{i_j}|v\rangle$ according to their magnitude and remove the least important basis vectors.
4. Replace the discarded basis vectors by new basis vectors. These are selected at random from a pool of candidate basis vectors which are connected to the old basis vectors through non-vanishing matrix elements of H.
5. Redefine S as the subspace spanned by the updated set of basis vectors and repeat steps 2 through 5.

If the subset of basis vectors is sufficiently large, the exact eigenvectors will be stable fixed points of the update process.

The purpose of this short tutorial is to provide additional information for those interested in writing or using codes that implement QSE methods. We discuss the calculation of Hamiltonian matrix elements in the momentum Fock space representation and the generation and selection of new basis states. We also include a simple program which applies the QSE method to find the ground state of an input matrix. Readers interested in an introductory overview of QSE diagonalization are encouraged to consult the references in [1].

2 Fock states and the Hamiltonian matrix

It is most effective to describe the details of a QSE calculation in the context of a specific example. In this discussion we consider ϕ^4 theory in $1+1$ dimensions, the same example as in [1]. Some of the techniques described here have been specifically optimized for scalar field theories. Other systems require a somewhat different set of tools and will be discussed in the future.

The Hamiltonian density for ϕ^4 theory in $1+1$ dimensions has the form

$$\mathcal{H} = \tfrac{1}{2}\left(\tfrac{\partial\phi}{\partial t}\right)^2 + \tfrac{1}{2}\left(\tfrac{\partial\phi}{\partial x}\right)^2 + \tfrac{\mu^2}{2}\phi^2 + \tfrac{\lambda}{4!}{:}\phi^4{:},$$

where the normal ordering is with respect to the mass μ. We consider the system in a periodic box of length $2L$. We then expand in momentum modes and reinterpret the problem as an equivalent Schrödinger equation [2]. We will use a momentum cutoff scheme where the modes q_n are restricted in momentum so that $|n| \leq N_{max}$. The resulting Hamiltonian is

$$\begin{aligned} H = &-\tfrac{1}{2}\sum_n \tfrac{\partial}{\partial q_{-n}}\tfrac{\partial}{\partial q_n} + \tfrac{1}{2}\sum_n \left(\omega_n^2(\mu) - \tfrac{\lambda b(\mu)}{4L}\right) q_{-n}q_n \\ &+ \tfrac{\lambda}{4!2L}\sum_{n_1+n_2+n_3+n_4=0} q_{n_1}q_{n_2}q_{n_3}q_{n_4}, \end{aligned} \tag{1}$$

where

$$\omega_n(\mu) = \sqrt{\tfrac{n^2\pi^2}{L^2} + \mu^2} \tag{2}$$

and $b(\mu)$ is the coefficient for the mass counterterm

$$b(\mu) = \sum_n \tfrac{1}{2\omega_n(\mu)}. \tag{3}$$

It is convenient to split the Hamiltonian into free and interacting parts with respect to an arbitrary mass μ':

$$H_{free} = -\tfrac{1}{2}\sum_n \tfrac{\partial}{\partial q_{-n}}\tfrac{\partial}{\partial q_n} + \tfrac{1}{2}\sum_n \omega_n^2(\mu')\, q_{-n}q_n, \tag{4}$$

$$H = H_{free} + \tfrac{1}{2}\sum_n \left(\mu^2 - \mu'^2 - \tfrac{\lambda b(\mu)}{4L}\right) q_{-n}q_n + \tfrac{\lambda}{4!2L}\sum_{n_1+n_2+n_3+n_4=0} q_{n_1}q_{n_2}q_{n_3}q_{n_4}. \tag{5}$$

μ' will be used to define the basis states of our Fock space. Since H is independent μ', it is useful to perform calculations for different μ' to obtain an estimate of the error. It is also helpful to find the range of values for μ' which maximizes the quasi-sparsity of the eigenvectors. This can be used to improve the accuracy of the QSE calculation.

Let us define ladder operators with respect to μ',

$$a_n(\mu') = \tfrac{1}{\sqrt{2\omega_n(\mu')}}\left[q_n\omega_n(\mu') + \tfrac{\partial}{\partial q_{-n}}\right] \tag{6}$$

$$a_n^\dagger(\mu') = \tfrac{1}{\sqrt{2\omega_n(\mu')}}\left[q_{-n}\omega_n(\mu') - \tfrac{\partial}{\partial q_n}\right]. \tag{7}$$

These satisfy the usual commutation relations,

$$\left[a_n, a_m^\dagger\right] = \delta_{nm} \tag{8}$$

$$[a_n, a_m] = \left[a_n^\dagger, a_m^\dagger\right] = 0. \tag{9}$$

The Hamiltonian can now be rewritten as

$$H = \sum_n \omega_n(\mu')a_n^\dagger a_n + \tfrac{1}{4}(\mu^2 - \mu'^2 - \tfrac{\lambda b(\mu)}{4L})\sum_n \tfrac{\left(a_{-n}+a_n^\dagger\right)\left(a_n+a_{-n}^\dagger\right)}{\omega_n(\mu')} + \tfrac{\lambda}{192L}\sum_{n_1+n_2+n_3+n_4=0}\left[\tfrac{\left(a_{n_1}+a_{-n_1}^\dagger\right)}{\sqrt{\omega_{n_1}(\mu')}}\tfrac{\left(a_{n_2}+a_{-n_2}^\dagger\right)}{\sqrt{\omega_{n_2}(\mu')}}\tfrac{\left(a_{n_3}+a_{-n_3}^\dagger\right)}{\sqrt{\omega_{n_3}(\mu')}}\tfrac{\left(a_{n_4}+a_{-n_4}^\dagger\right)}{\sqrt{\omega_{n_4}(\mu')}}\right]. \tag{10}$$

In (10) we have omitted constants contributing only to the vacuum energy.

We can represent any momentum-space Fock state as a string of occupation numbers,

$$\left|o_{-N_{\max}}, \cdots, o_{N_{\max}}\right\rangle, \tag{11}$$

where

$$a_n^\dagger a_n \left|o_{-N_{\max}}, \cdots, o_{N_{\max}}\right\rangle = o_n \left|o_{-N_{\max}}, \cdots, o_{N_{\max}}\right\rangle. \tag{12}$$

From the usual ladder operator relations, it is straightforward to calculate the matrix element of H between two arbitrary Fock states. In [1] auxiliary cutoffs were used to render the dimension of Fock space finite. This is in fact an unnecessary restriction and QSE diagonalization encounters no difficulties dealing with the full infinite-dimensional space.

3 New basis vectors

Aside from calculating matrix elements, the only other fundamental operation involving basis vectors is the generation of new basis vectors. As mentioned before, the new states should be connected to an old basis vector through non-vanishing matrix elements of H. Let us refer to the old basis vector as $|e\rangle$. For this example there are two types of terms in our interaction Hamiltonian, a quartic interaction

$$\sum_{n_1,n_2,n_3} \left(a_{n_1} + a^\dagger_{-n_1}\right)\left(a_{n_2} + a^\dagger_{-n_2}\right)\left(a_{n_3} + a^\dagger_{-n_3}\right)\left(a_{-n_1-n_2-n_3} + a^\dagger_{n_1+n_2+n_3}\right), \tag{13}$$

and a quadratic interaction

$$\sum_n \left(a_{-n} + a^\dagger_n\right)\left(a_n + a^\dagger_{-n}\right). \tag{14}$$

To produce a new vector from $|e\rangle$ we simply choose one of the possible operator monomials

$$\begin{aligned} &a_{n_1}a_{n_2}a_{n_3}a_{-n_1-n_2-n_3},\ a^\dagger_{-n_1}a_{n_2}a_{n_3}a_{-n_1-n_2-n_3}, \cdots, \\ &a_{-n}a_n,\ a^\dagger_n a^\dagger_{-n}, \cdots \end{aligned} \tag{15}$$

and let it act upon $|e\rangle$. Our experience is that the interactions involving the small momentum modes are generally more important than those for the large momentum modes, a signal that the ultraviolet divergences have been properly renormalized. For this reason it is best to arrange the selection probabilities such that the smaller values of $|n_1|$, $|n_2|$, $|n_3|$ and $|n|$ are chosen more often.

We note that the new basis vector selection will occasionally fail. Either the vector is zero due to an annihilation operator acting on an unoccupied state or the vector is already in our subset of basis vectors and therefore not new. In either case we simply select a new basis vector again. If the selection process fails repeatedly then a different old basis vector $|e'\rangle$ is used.

4 Sample code

We have considered two basic operations, the calculation of matrix elements and the generation of new basis states. The subroutines which perform these tasks will depend on the form of the quantum field Hamiltonian. The main structure of the QSE algorithm, however, is independent of the details of the Hamiltonian. We demonstrate the essential features of the algorithm with the following short MATLAB program. In this example we find the ground state of a finite matrix. To keep the example as simple as possible, we avoid calculating matrix elements and generating new basis states by assuming that the entire Hamiltonian matrix can be stored in memory.

```
niter = 30;
nactive = 100;
nretain = 80;
H = loadsparse('samplematrix');
vecs = [1:nactive];
for iter = 1:niter
    [v,d] = eigs(H(vecs,vecs),'SR',1);
    weights1 = v.^2;
    [sortelements, sortorder] = sort(-weights1);
    vecs = vecs(sortorder(1:nretain));
    wtsum1 = cumsum(weights1)/sum(weights1);
    while (length(vecs) <nactive)
        over1 = find(rand*wtsum1(nretain) <wtsum1);
        seedvec = vecs(over1(1));
        candidates = find(H(:,seedvec));
        candidates = candidates(candidates ~= seedvec);
        weights2 = abs(H(candidates,seedvec));
        wtsum2 = cumsum(weights2)/sum(weights2);
        over2 = find(rand <wtsum2);
        newvec = candidates(over2(1));
        vecs = unique([vecs newvec]);
    end
end
save results.mat vecs v d
```

`niter` is the number of iterations of the algorithm. `nactive` is the number of basis states used to form the subspace S, and `nretain` is the number of basis states kept after removing the tail of the eigenvector distribution. `loadsparse` is a program which reads in a datafile called `samplematrix`. This

data is stored in the matrix variable `H`. The remaining code can be broken down in terms of the five basic steps of the QSE algorithm.

Select a subset of orthonormal basis vectors $\{e_{i_1}, \cdots, e_{i_n}\}$ *and call the corresponding subspace* S.

```
vecs = [1:nactive];
```

The variable `vecs` is a row vector which labels the basis vectors in the subspace S. In our example `vecs` is initialized to be the first `nactive` basis vectors. We are assuming that the ground state of the system has a non-zero overlap with at least one of the first `nactive` basis vectors. A more general initialization would be to choose a random subset of basis vectors.

Diagonalize H *restricted to* S *and find one eigenvector* v.

```
[v,d] = eigs(H(vecs,vecs),'SR',1);
```

The function `eigs` is a sparse matrix Arnoldi diagonalization routine. It is being called with the parameters `'SR',1` which tells `eigs` to find the eigenvalue and eigenvector with the smallest real part. The Hamiltonian submatrix corresponding with basis vectors in `vecs` is diagonalized. The eigenvector is stored in the column vector `v` and the eigenvalue is stored as the variable `d`.

Sort the basis components $\langle e_{i_j}|v\rangle$ *according to their magnitude and remove the least important basis vectors.*

```
weights1 = v.^2;
[sortelements, sortorder] = sort(-weights1);
vecs = vecs(sortorder(1:nretain));
```

`weights1` is a column vector which contains the squares of the components of the eigenvector `v`. `sortelements` is a list of the elements of `-weights1` from smallest to greatest. `sortorder` is the corresponding list of locations for these elements. If for example `weights1 = [.2 .5 .3]`, then `sortorder = [2 3 1]`. The list `sortelements` is not used. Only the `nretain` most important basis vectors are kept in `vecs`.

Replace the discarded basis vectors by new basis vectors. These are selected at random from a pool of candidate basis vectors which are connected to the old basis vectors through non-vanishing matrix elements of H.

```
wtsum1 = cumsum(weights1)/sum(weights1);
while (length(vecs) <nactive)
    over1 = find(rand*wtsum1(nretain) <wtsum1);
    seedvec = vecs(over1(1));
    candidates = find(H(:,seedvec));
    candidates = candidates(candidates ~= seedvec);
    weights2 = abs(H(candidates,seedvec));
```

```
        wtsum2 = cumsum(weights2)/sum(weights2);
        over2 = find(rand <wtsum2);
        newvec = candidates(over2(1));
        vecs = unique([vecs newvec]);
    end
```

`cumsum` is a function which takes the input list and creates another list consisting of partial sums. `wtsum1` is therefore an ascending string of numbers between 0 and 1 such that the difference between an entry and its preceding entry is proportional to the corresponding value of `weights1`. `over1` is a location list of elements of `wtsum1` which are greater than the product of a generated random number between 0 and 1 and `wtsum1(nretain)`. `over1(1)` is the first entry of `over1`. We note that `over1(1)` must be less than or equal to `nretain`. `seedvec` is the basis vector which corresponds with `over1(1)`. `candidates` is a column vector consisting of all basis vectors which have a non-zero matrix element with `seedvec`. Since the task is to find new basis vectors, `seedvec` is removed from `candidates`.

`weights2` is a column vector containing the absolute values of the matrix transition elements from `seedvec`. `wtsum2` is an ascending string between 0 and 1 such that the difference between an entry and its preceding entry is proportional to the value of `weights2`. `over2` is a location list of elements of `wtsum2` greater than a generated random number. `newvec` is the basis vector which corresponds with `over2(1)`. `newvec` is appended to the set `vecs`. The function `unique` sorts the input list and removes any repetitions. This loop is repeated until `vecs` has `nactive` elements.

Redefine S as the subspace spanned by the updated set of basis vectors and repeat steps 2 through 5.

```
    for iter = 1:niter
        ...
    end
    save results.mat vecs v d
```

The algorithm is iterated `niter` times and the final results are saved in the file `results.mat`. More examples of QSE diagonalization codes for various field theory systems written in MATLAB, Fortran, and/or C++ will be made available in the future.

5 Summary

In this short tutorial we have listed some supplemental material for those interested in writing or using codes that implement QSE methods. We have

discussed the calculation of Hamiltonian matrix elements and the selection of new basis states. We have also presented a simple MATLAB program which applies the QSE method to find the ground state of a matrix.

The discussion here was limited to the basic QSE method. As presented in the Guangzhou workshop there have also been interesting new developments in QSE diagonalization regarding the technique of stochastic error correction. This technique will be presented in a forthcoming work [4].

Acknowledgments

I thank my collaborators on the papers cited here and the organizers and participants of the International Workshop on Nonperturbative Methods and Lattice QCD in Guangzhou. Financial support was provided by the National Science Foundation.

References

1. D. J. Lee, N. Salwen, D. D. Lee, hep-th/0002251; D. Lee, hep-th/0003296.
2. P. Marrero, E. Roura, D. Lee, Phys. Lett. B471 (1999) 45.
3. N. Salwen, D. Lee, Phys. Rev. D62 (2000) 025006.
4. D. Lee, N. Salwen, M. Windoloski, work in progress.